

Combining One-Class Classifiers via Meta-Learning

Eitan Menahem Lior Rokach and Yuval Elovici

{EITANME, LIORRK, ELOVICI}@BGU.AC.IL

Department of Information Science Engineering and Deutsche Telekom Laboratories
Ben Gurion University, Be'er Sheva, 84105, Israel

Abstract. We examine various methods for combining the output of one-class models. In particular, we show that simple meta-learning based ensemble achieves better result than weighting methods. Furthermore we propose a new one-class ensemble scheme, called TUPSO that uses meta-learning for combining multiple one-class classifiers. We also present a new one-class classification performance measures to weigh the base-classifiers, a process that proved helpful for increasing the classification performance of the induced ensemble. Our experimental study shows that the proposed method significantly outperforms exiting methods.

1 Introduction and Background

One-class classification aims to differentiate instances of class of interest from all other instances. The one-class classifier is trained from a training set containing only the instances of that class. Many one-class classification algorithms have been investigated [1, 2]. The variety of classification algorithms is both positive and negative. On the one hand, there are plenty of techniques to choose from, but on the other hand, choosing the right one can be difficult. This is because evaluating one-class classifiers performance is problematic. By definition, the data collections contain only one-class examples, and thus performance metrics, such as false-positive (FP) and true negative (TN), cannot be computed. In the absence of FP and TN , derived performance metrics, such as classification accuracy, precision, AUC and others cannot be computed. One way to address this difficulty is to use an ensemble of classifiers as opposed to choosing the best classifier. The main idea behind the ensemble methodology is to weigh several individual classifiers and combine them in order to obtain a classifier that outperforms them all. Indeed, previous works in supervised ensemble learning show that combining classification models can produce a better classifier in terms of prediction accuracy [3].

Compared to supervised-learning, research in the one-class ensemble research is limited [4]. In particular, only one combining method, the Fix-rule ensemble, was considered for one-class ensemble [5, 6, 7]. In this method, the combiner regards each participating classifier's output as a single vote, upon which it applies some aggregative function (a combining rule) to produce a final classification. Henceforth we use the notation $I(\cdot)$ as the indicator function, $P_k(x|\omega_{T_c})$ as the estimated probability of instance x given the target class ω_{T_c} , $f_{T,k}$ as the fraction of the target class that should be accepted for classifier k and θ_k notates

the classification threshold for classifier k . A list of fixed combining rules is presented in Table 1.

The fixed rule ensemble techniques, however, are not optimal, as the rules are

Combining Rule	Combination Rule Formula
Majority voting (Voting)	$y(x) = I_{\geq k/2}(\sum_k I(P_k(x \omega_{T_c}) \geq \theta_k))$
Mean vote	$y(x) = \frac{1}{r} \sum_k I(P_k(x \omega_{T_c}) \geq \theta_k)$
Mean weighted vote	$y(x) = \frac{1}{r} \sum_k [f_{T,k} I(P_k(x \omega_{T_c}) \geq \theta_k) + (1 - f_{T,k}) I(P_k(x \omega_{T_c}) < \theta_k)]$
Average rule	$y(x) = \frac{1}{r} \sum_k P_k(x \omega_{T_c})$
Max rule	$y(x) = \text{argmax}_k [P_k(x \omega_{T_c})]$
Product rule	$y(x) = \prod_k [P_k(x \omega_{T_c})]$
Exclusive voting rule	$y(x) = I_1(\sum_k I(P_k(x \omega_{T_c}) \geq \theta_k))$
Weighted votes product	$y(x) = \frac{\prod_k [f_{T,k} I(P_k(x \omega_{T_c}) \geq \theta_k)]}{\prod_k [f_{T,k} I(P_k(x \omega_{T_c}) \geq \theta_k)] + \prod_k [(1 - f_{T,k}) I(P_k(x \omega_{T_c}) < \theta_k)]}$

Table 1: Fix combining rules.

usually assigned statically and independent of the training data. In order to find better ensemble solutions, researchers tend to import ideas from other learning domains. Unfortunately, the absence of negative examples in the training-sets of one-class problems make the adoption of many fine ensemble schemes difficult, especially with those who rely on some performance metrics, e.g. Weighting Performance and Grading. The ensemble method we pursue here, TUPSO, is based on meta-learning, aided by one-class performance evaluator. We show this method is indeed practical and worthwhile for one-class ensemble learning. Moreover, it significantly outperforms the Fix-rule technique.

2 Meta-Learning-Based Ensembles for One-Class Problems

The proposed ensemble method, TUPSO, is intended to function in one-class scenarios where multiple, and possibly diverse, classifiers exist. Its main task is to combine one-class base-classifiers via meta-classifier. TUPSO is aided by a heuristic performance evaluator which estimates the classification performance of each of the base-classifiers and output a performance vector, $Perf_{vect} = \{Pref_1, \dots, Pref_m\}$. The estimated performance is then translated into static weights, α_i , which the meta-learning algorithm uses. $\alpha_i = Perf_i * 1 / \sum_{j=1}^m Perf_j$, $\forall i = 1 \dots m$, where m is the number of instance.

TUPSO ensemble, as shown in Figure 1, is made up of four major components: (1) Base-Classifiers, (2) Performance Evaluator, (3) Aggregate Features Extractor and (4) Meta Classifier.

One of the building blocks of meta-learning is the meta-features, which are measured properties of the base-classifier(s). A collection of meta-features makes a meta-instance, which is used for training the meta-classifier and upon which the trained meta-classifier produces the ensemble prediction. The Aggregate Features Extractor produces meta-features by using multiple aggregations of the

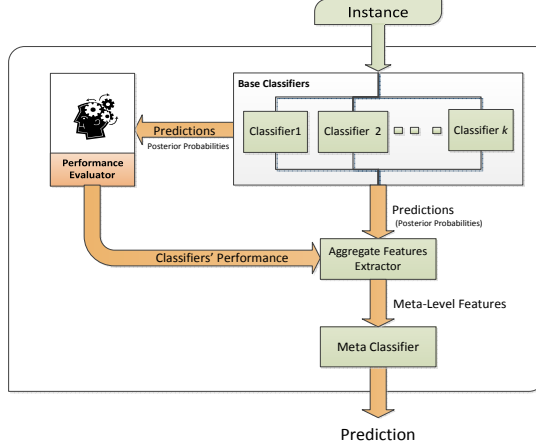


Fig. 1: The TUPSO ensemble scheme.

base-classifiers outputs. Let $P_m = \langle p_{m_1}, \dots, p_{m_k} \rangle$ be the vector containing the base-classifiers outputs p_{m_1}, \dots, p_{m_k} , where k is the number of base-classifiers. In Table 2 we define six such aggregate meta-features $F_{aggr} = \{f_1, \dots, f_6\}$. During the training phase, the meta-features are applied for training the meta-classifier. In the on-line phase they are used by the meta-classifier that produces the final prediction.

Feature Name	Abbreviation	Feature Definition
Sum of Votes	$SumV$	$f_1(P_m) = \sum_{i=1}^k 1_{\{p_{m_i} \geq 0.5\}}(P_{m_i})$
Sum of Predictions	$SumP$	$f_2(P_m) = \sum_{i=1}^k P_{m_i}$
Sum of weighted Predictions	$SumWP$	$f_3(P_m) = \sum_{i=1}^k \alpha_i * P_{m_i}$
Variance of Votes	$VarV$	$f_4(P_m) = Var(1_{\{p_{m_i} \geq 0.5\}}(P_{m_i}))$
Variance of Predictions	$VarP$	$f_5(P_m) = Var(P_m)$
Variance of weighted Predictions	$VarWP$	$f_6(P_m) = Var(\alpha * P_m)$

Table 2: Aggregate features, produced by the Aggregate Features Extractor module

3 Estimating the Performance of One-Class Classifiers

The base-classifiers performance is a key factor for producing effective aggregate meta-features. Unfortunately, the inherent absence of negative examples makes the performance assessing difficult. This is because the two important values, the false positive (FP) and true negative (TN) rates, cannot be measured, and thus, calculating some performance metrics such as Accuracy, Precision and F -score becomes impossible. Notice that each of the following performance metrics misses one or more values: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, $Precision = \frac{TP}{TP+FP}$

and $F\text{-score} = \frac{2PR}{P+R}$, where P is the Precision and R is the Recall. Alternatively, we propose using a one-class approach for estimating classifiers performance. A criterion proposed in [8] estimates the $F\text{-score}$ by using classifier's prediction on positive instances. The criterion, $\frac{rp}{Pr[Y=+1]}$, shares some characters with the $F\text{-Score}$; note that $Pr[f(x) = +1|Y = +1]Pr[Y = +1] = Pr[Y = +1|f(x) = +1]Pr[f(x) = 1] \Leftrightarrow Pr[f(x) = +1|Y = +1]/Pr[f(x) = +1] = Pr[Y = +1|f(x) = +1]/Pr[Y = +1] \Leftrightarrow \frac{r}{Pr[f(x)=+1]} = \frac{p}{Pr[Y=+1]}$. By multiplying both sides by r , we get the desired measure, henceforth denoted as Positive-Only $F\text{-score}$ (Fix-rule): $POF \equiv \frac{r^2}{Pr[f(x)=+1]}$. Notice that the recall $r = Pr[f(x) = +1|y = +1]$ can be estimated from the classifier predictions on positive labeled examples, and $Pr[f(x) = 1]$ can be estimated with $Pr[f(x) = +1] = m^{(-1)} * \sum_{i=1}^m f(x) = +1$ from the classifier output on the validation set.

4 Evaluation

In this section we examine two aspects concerning TUPSO. First, we test the proposed aggregation functions under various datasets. Secondly, we test how well TUPSO performs in comparison to some common Fix-rule methods. In order to estimate the generalized classification performance of the mentioned ensemble schemes, a 5x2 cross-validation procedure was performed [9].

4.1 Experiment Setup

The evaluation of TUPSO is consist of three dimensions: Datasets (12 popular datasets from the UCI collection [10]), combining method (TUPSO or fixe-rule method) and base-classifier performance metric (POF or none). We used six inducer setups (i.e. base-classifiers), induced by four algorithms: (i) ADIFA-HM and (ii) ADIFA-GM [11], (iii) GDE [12], (iv) PGA [13], (v) $OC\text{-}SVM_1$ and (vi) $OC\text{-}SVM_2$ [1]. The base-classifiers properties were kept static during the entire evaluation. Afore mentioned base-classifiers were selected as they represent three prominent families of one-class classifiers: density (ADIFA) nearest-neighbor (GDE and PGA) and boundary (SVM). Table 3 present the setup parameters.

Base Classifier	Algorithm	Parameters
ADIFA-HM	<i>ADIFA</i>	<i>Aggregation = HarmonicMean, Sensitivity = 2%</i>
ADIFA-GM	<i>ADIFA</i>	<i>Aggregation = GeometricMean, Sensitivity = 1%</i>
OC-GDE	<i>OC-GDE</i>	<i>n/a</i>
OC-PGA	<i>OC-PGA</i>	<i>k = 3 (3 - nearestneighbor), $\alpha = 0.01$</i>
OC-SVM ₁	<i>OC-SVM</i>	<i>kernel = linear, $\nu = 0.05$</i>
OC-SVM ₂	<i>OC-SVM</i>	<i>kernel = polinomial, $\nu = 0.05$</i>

Table 3: Base-Classifiers setup parameters. Shown are the non-default parameters.

For the Mean-Vote rule we used the classification cutoff $\theta_k = 0.75$.

4.2 Measured Metrics

For evaluating the *real* performance of both individual classifiers and ensemble methods we used a two-class performance metric, rather than *POF*. In contrast to the training process, in which the absence of negative examples dictates the use of one-class evaluation metrics (i.e. *POF*), the dataset in the testing phase included instances of both classes, and thus allows the use two-class performance measure. Specifically, we used the Area under the ROC curve (*AUC*).

4.3 Experimental Results

In Table 4 we present the evaluation results. We use 'All', 'WF' and 'Non-WF' to denote all-, weighted- and non-weighted- aggregative features respectively (see Table 2). Inside the parenthesis is the AUC rank of the corresponding ensemble method.

Comb.	TUPSO								Fix-Rule			
	All	WF	Non-WF	SumP	SumV	SumWP	VarP	VarWP	Voting	Max	MeanV	Product
Balance-Scale	0.83(4)	0.85(1)	0.82(6)	0.83(5)	0.85(2)	0.84(3)	0.69(9)	0.73(7)	0.67(12)	0.51(13)	0.68(11)	0.69(10)
Ecoli	0.91(7)	0.92(4)	0.91(5)	0.94(3)	0.94(2)	0.94(1)	0.79(10)	0.44(13)	0.91(6)	0.78(11)	0.84(9)	0.9(8)
Disease	0.69(9)	0.75(2)	0.7(8)	0.73(3)	0.75(1)	0.71(5)	0.68(10)	0.7(7)	0.61(12)	0.51(13)	0.7(6)	0.71(4)
Ionosphere	0.96(2)	0.96(3)	0.96(1)	0.96(5)	0.96(6)	0.96(4)	0.51(13)	0.7(10)	0.89(7)	0.76(8)	0.65(11)	0.74(9)
Letter	0.97(2)	0.97(1)	0.97(3)	0.97(5)	0.96(6)	0.97(4)	0.88(9)	0.93(7)	0.82(11)	0.5(13)	0.78(12)	0.9(8)
M-feat	0.96(5)	0.96(6)	0.96(4)	0.97(2)	0.97(3)	0.97(1)	0.78(10)	0.62(13)	0.91(7)	0.63(12)	0.85(9)	0.85(8)
Opti-Digits	0.98(2)	0.98(1)	0.98(3)	0.97(4)	0.97(5)	0.97(6)	0.78(10)	0.61(13)	0.86(8)	0.65(12)	0.78(9)	0.9(7)
Page-blocks	0.93(5)	0.92(6)	0.94(1)	0.94(3)	0.94(2)	0.93(4)	0.8(10)	0.83(9)	0.85(8)	0.52(13)	0.74(11)	0.87(7)
Splice	0.96(3)	0.96(2)	0.95(4)	0.95(5)	0.94(8)	0.95(6)	0.95(7)	0.96(1)	0.56(12)	0.5(13)	0.77(11)	0.87(10)
Vote	0.93(3)	0.93(5)	0.93(4)	0.93(2)	0.93(6)	0.94(1)	0.75(9)	0.71(10)	0.76(8)	0.51(13)	0.71(11)	0.82(7)
Breast-Cancer	0.96(3)	0.97(1)	0.96(2)	0.96(5)	0.96(6)	0.96(7)	0.8(10)	0.87(8)	0.96(4)	0.5(13)	0.75(12)	0.84(9)
Zoo	0.82(4)	0.88(1)	0.82(5)	0.82(3)	0.78(7)	0.83(2)	0.69(10)	0.78(9)	0.67(11)	0.5(13)	0.78(8)	0.8(6)
Avg. Rank	4.1	2.8	3.8	3.8	4.5	3.7	9.8	8.9	8.8	12.3	10	7.8

Table 4: TUPSO vs. Fix-Rule AUC result table.

We can see that the weighted features (*SumWP* and *VarWP*) together produced the best performance. It seems that mixing together the non-weighted and the weighted features produces a marginally weaker ensemble.

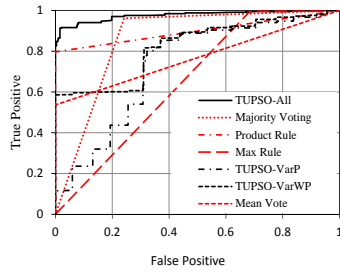


Fig. 2: ROC Curves of TUPSO and Fix-rule on *Opt-digits* dataset

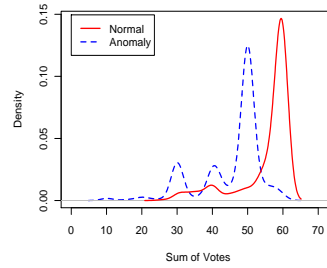


Fig. 3: Density graph for the SumV aggregate feature for *Letter* dataset.

In Figure 2 we present the ROC graphs of the tested ensemble methods generated using the Optical-Digits dataset. The best curve belongs to a TUPSO version that uses all the aggregate features defined in Table 2. A weakness of the Fix-rule technique is demonstrated in Figure 3. Having 60 base-classifiers,

we plotted the density of the sum of votes for both the 'normal' and 'anomaly' classes. The classification cutoff value which produces the least classification errors is around 55. Voting, however, will classify any sum of votes above 30 as 'normal' and since most of 'anomaly' instances receive more than 30 votes, using it will result in a high FNR and a low TPR. The Max-rule will perform equally poorly because the probability that no base-classifier will vote "normal" is very low for 'anomaly' instances. TUPSO, in contrast, will assign a low probability for sum of votes lower than 55 and hence classify instances more proficiently.

5 Conclusions and Future Work

In this paper we proposed a new meta-learning based ensemble scheme for one-class problems. The ensemble scheme learns a combining function upon aggregates of the base-classifiers' predictions. To improve the aggregates inductive power, we implemented a classification performance evaluator, which we found very effective. Further on, we would like to examine TUPSO on more datasets and investigate additional one-class performance evaluators. Finally, we would like to discover how TUPSO performs compared to the ensemble's best-classifier.

References

- [1] Chris M. Bishop. Novelty detection and neural network validation, 1994.
- [2] Aryeh Kontorovich, Danny Hendler, and Eitan Menahem. Metric anomaly detection via asymmetric risk minimization. In *SIMBAD*, pages 17–30, 2011.
- [3] Eitan Menahem, Lior Rokach, and Yuval Elovici. Troika - an improved stacking schema for classification tasks. *Inf. Sci.*, 179(24):4097–4122, 2009.
- [4] Giorgio Giacinto, Roberto Perdisci, Mauro Del Rio, and Fabio Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008.
- [5] David M.J. Tax and Robert P.W. Duin. Combining one-class classifiers. In *in Proc. Multiple Classifier Systems, 2001*, pages 299–308. Springer Verlag, 2001.
- [6] Piotr Juszczak and Robert P. W. Duin. Combining one-class classifiers to classify missing data. In *Multiple Classifier Systems*, pages 92–101, 2004.
- [7] Santi Seguí, Laura Igual, and Jordi Vitrià. Weighted bagging for graph based one-class classifiers. In *MCS*, pages 1–10, 2010.
- [8] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, pages 448–455, 2003.
- [9] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.
- [10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
- [11] E. Menahem, L. Roakach, and Y. Elovici. Anomaly detection via attribute distribution function approximation, 2011.
- [12] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*. Kluwer, 2002.
- [13] Edwin M. Knorr and Raymond T. Ng. A unified notion of outliers: Properties and computation. In *In Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 219–222. AAAI Press, 1997.